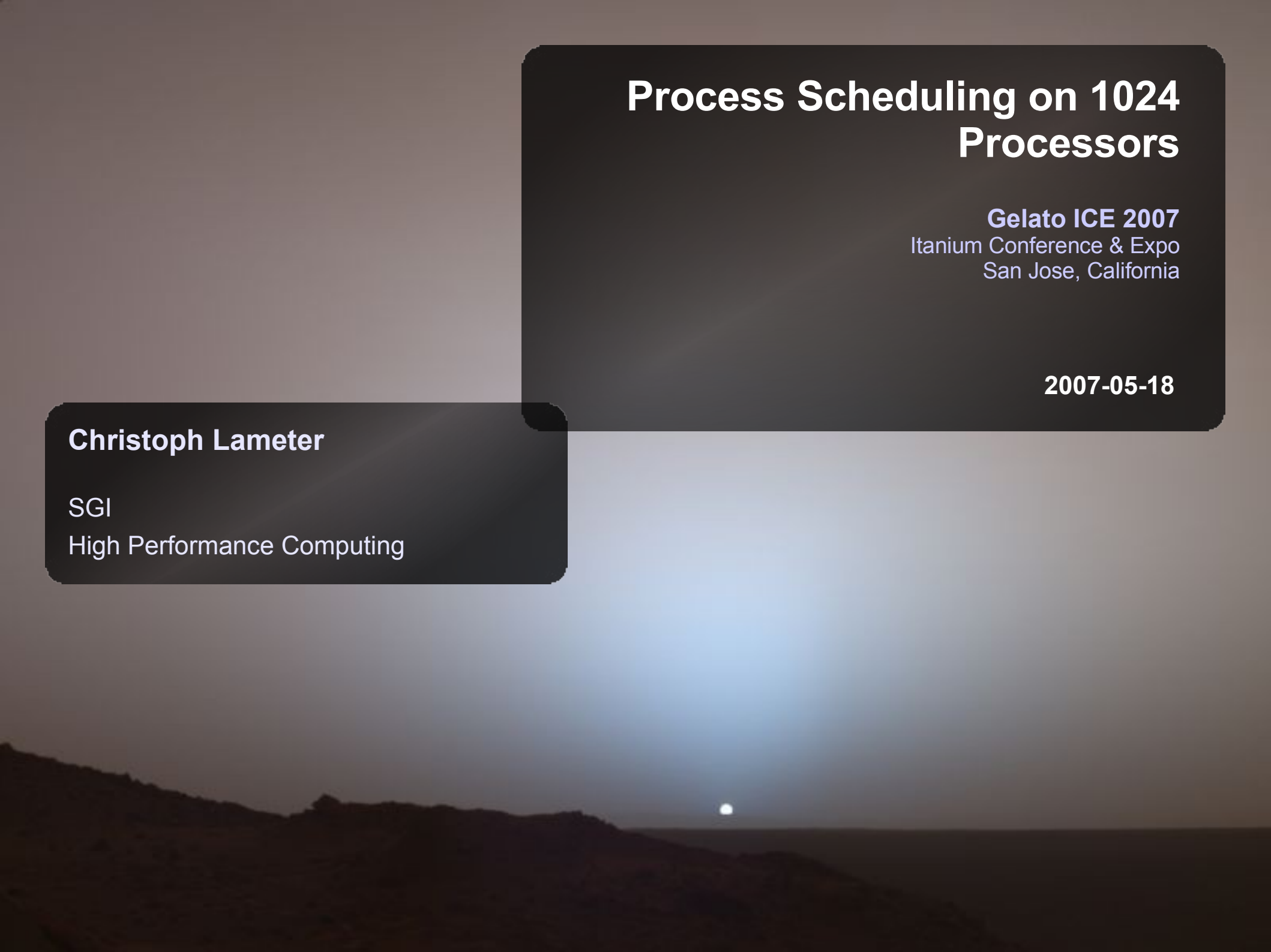


Process Scheduling on 1024 Processors

Gelato ICE 2007
Itanium Conference & Expo
San Jose, California

2007-05-18

Christoph Lameter

SGI
High Performance Computing

Overview

- Linux Scheduler basics
- Process migration
  - Scheduling domain
  - Load balancing
- Issues
  - SMP scheduler on NUMA
  - Scheduler freezing
  - Livelocks etc
  - Load balancing overhead
- Solutions
  - Placing memory
  - Load balancing with interrupts enabled
  - Avoiding frequent large scale load balancing
- Conclusion

The Linux Scheduler

- Ingo Molnar's design of an O(1) scheduler
  - Processes in run queues
    - Active
    - Expired
    - Active empty -> Expired becomes the active one.
  - Load Balancing
    - Necessary for SMP to utilize full power of all processors
    - Trivial for a 2 or so processor SMP system for which the scheduler seems to have been designed.
    - More complex for larger systems
    - Trade off: Processor caches are cold if process is moved to new processor.

SMP scheduler for NUMA

- Scheduler only considers caching effects
- We extrapolate from SMP features to realize scheduling behavior on NUMA
- In practice many users of NUMA explicitly control processor placement.
- General assumption of having a small number of processors (small meaning in single digits)
- NUMA aware load balancing would have to consider page placement of a process. It is better to move a process that has pages on the remote node.
- Without that data we simply assume that moving over long distances is more expensive.

Scheduler Load Balance Freeze Bug

- Scheduler must be O(1)
- We make only one attempt to load balance
- If it fails too bad
- It fails if the busy processor has processes that cannot be moved due to processor affinity
- If that processor is the busiest processor then Load balancing will stop for good
- Solution: Retry rebalance without considering the busy processor.
- Bug still exists in SLES9. Fixed in SLES10

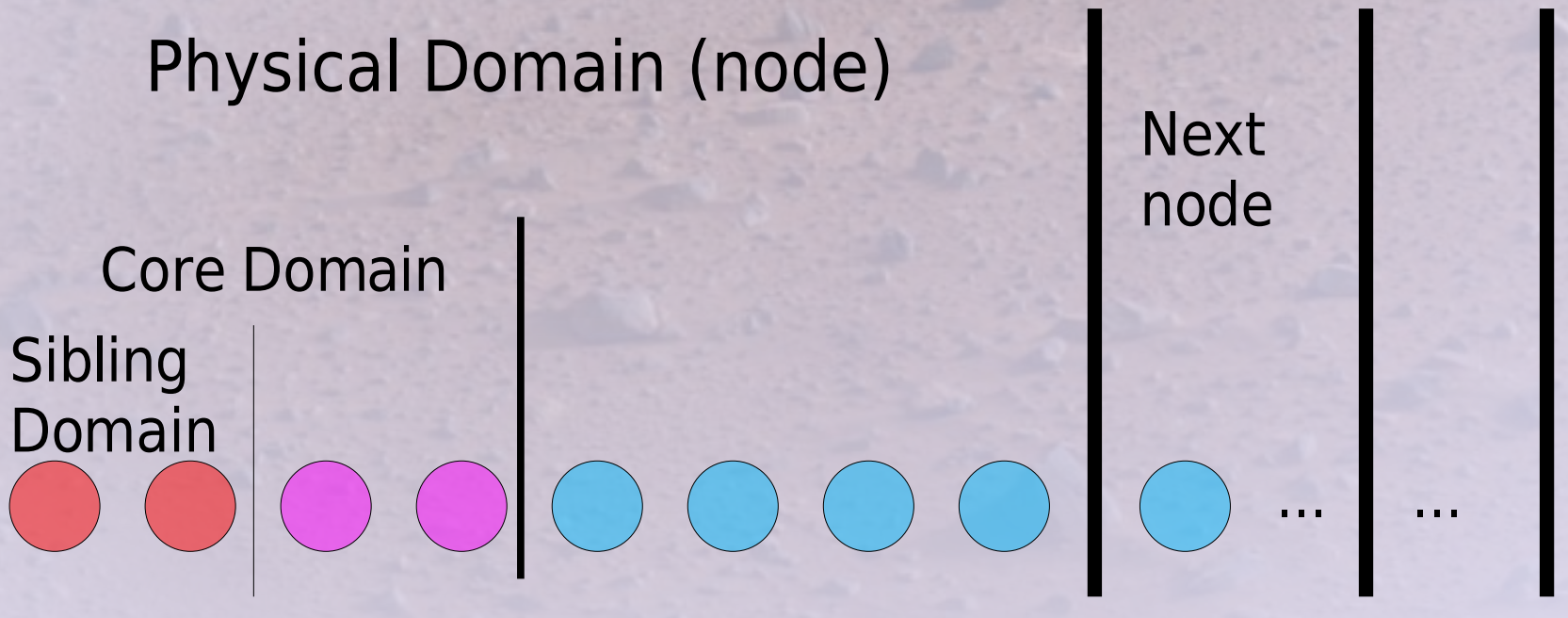
Sched Domains

- Load balancing in NUMA system is a complex given that the cost of migration a process is dependent on the dispersal of the pages over nodes.
- Sched Domains designed to solve the issues of load balancing in large systems (but conceptual clash with cpusets etc)
- Hierarchy of domains from single package, to processor, to node, to 16 node chunk and then to all nodes in the system.
- Allnodes: Covering all processors on all nodes
- Node group: Covers 16 neighboring nodes
- Phys domain: A single node
- Core domain: A single processor
- Sibling domain: Cores of a processor

Sched Domains Diagram

All nodes scheduling domain (1024 processors....)

Nodes Group (16 neighboring nodes)



Schedule Domain Characteristics

- Balance Interval
- Cache hot characteristics
- Flags
- span of domain
- busy factor
- Called from timer interrupt with interrupts disabled.
- Balance on idle
- Balance on exec
- Balance on fork
- Balance on wake
- Share cpu power
- Powersaving balance

Sched Domain issues

- Placement of sched domain data structures lead to remote fetches which are slow
- Load balancing with interrupts disabled from the timer interrupt.
- Concurrent large scale load balancing potential from lots of processors.
- Livelocks
- Timer interrupt may take longer than a single tick.

Fix 1: Balance only first processor in a Sched domain

- Solution by Suresh Siddha
- Reduces the number of load balances in particular for large domains where each processor used to perform large scale load balancing.
- On 1024 cpu system large scale load balance attempt are reduced by a factor of a thousand.
- This means that the first processor will draw processes which will then slowly be distributed over the rest of the processors using lower sched domains

Fix 2: Serialize large scale load balancing

- Insures that only one processor is scanning through all processor queues.
- Avoids overloading node with requests from remote processors
- Avoids load balance storms

Fix 3: Separate load balancing from time slice switching

- We do not always have to do load balancing.
- Time slice switching must always be done in timer interrupt
- Load balancing can be done differently and only be triggered if really necessary.
- Move load balancing out of timer interrupt to softirq.
- Softirq runs with interrupts enabled (IPIs are not held off, device interrupts can be serviced).

Fix 4: Data locality fixes

- Fix by Suresh Siddha
- Place all load balance data in per cpu area
- Prior load balance data for nodes and other supergroups was placed on a single node.
- Each cpu that load balanced had to reference that data frequently leading to potential bottlenecks.

Misc Fixes

- Remove useless staggering of load balancing

Conclusion

- I hope that load balancing has significantly improved on NUMA.
- First customer experiences in 2008 with SLES11 and RHEL6.
- New scheduler work mentioned by Andrew. But that is largely dealing with time slice management on a processor.
- Questions?